Physio Log: An application for easier collection of patient and test data for physiotherapists

Albert Ahnlide (BME20), Jesper Follin (BME20)

Sammanfattning—**When a stroke patient visits a physiotherapist for rehabilitation, various tests are often conducted. The results from these tests need to be stored somewhere, and currently, there is no efficient method for doing so. The best available option is an Excel spreadsheet that calculates the results, followed by the physiotherapist having to save the data in another way. This process is time-consuming, complex, and not particularly satisfying way of working. The aim of this project has been to develop a solution in the form of a software program where all patients' tests can be stored in one place. Additionally, the program provides new tools to the user for easily modifying patient data, visualizing results for patients, and conveniently adding new patients or selecting existing ones. The project has been carried out at Lunds Tekniska Högskola (LTH) and, in some cases, at the Health Science Center in Lund as well. We have been in contact with our supervisor, Håkan Carlsson, throughout the project. The program has been developed using the Python programming language and has been built as a GUI application with various windows for user navigation. Furthermore, a database has been created using SQL to store all patient and test data. The final application functions as intended, providing the user with an efficient and convenient way to input patient information and save test data. This saves time for physiotherapists and provides them with the means to present the patients with their results in a clear manner.**

I. Introduktion

STROKE är en allvarlig sjukdom och 2021 drabbades ungefär 25 400 personer och 5 900 avled på grund av just detta tillstånd [1]. Stroke är samlingsnamnet på det tillstånd som uppstår när en del av hjärnans nervvävnad påverkas av minskad syretillförsel till något område i hjärnan. Detta kan vara på grund av blodproppar eller hjärnblödning men den absolut vanligaste anledningen är hjärninfarkt. Symptomen för stroke kan vara svåra att upptäcka i tid då de oftast kommer väldigt snabbt. Det kan handla om förlamning, yrsel eller kraftig huvudvärk, svårigheter med talet eller att förstå vad andra säger [2]. Det kan även leda till försämrad fysisk förmåga och balans vilket fysioterapeuter kan hjälpa en att återfå. Stroke är därför en av de vanligaste anledningarna till att behöva besöka en fysioterapeut. Patienten får en bedömning av en fysioterapeut som innefattar patientens fysiska tillstånd inklusive styrka, rörlighet balans och koordination. Ett mål baserat på patientens fysiska tillstånd innan insjuknandet sätts upp och en plan görs för att komma tillbaks till den fysiska förmågan. Patienten får övningar att utföra antingen själv eller med hjälp av personal. Behandlingsplanen inkluderar olika övningar och tekniker som varierar beroende på patientens mål och behov. Det kan handla om att förbättra muskelstyrka i händer, få tillbaks balansen och koordinationen i kroppen eller öva upp gångförmågan och gånghastigheten. För att se hur patienten utvecklas över tid finns det olika tester som utförs hos fysioterapeuten. Exempel på dessa är 6 min walk test, där man ska gå så långt man kan på 6 minuter [3]. Ett annat är Grip and Pinch där patientens precisionsförmåga testas genom att ett objekt kläms fast på olika sätt av patienten [4]. Ett tredje är Box and Blocks. Här får patienten under 60 sekunder förflytta så många kuber hen kan från en låda till en annan [5]. Alla dessa används flitigt inom fysioterapi och är en viktig del för patientens rehabilitering. Informatoinen från dessa tester sparas ofta av fysioterapeuterna genom att använda initalerna från patienten. Denna data delas inte med andra och är ett simplare sätt att spara data än sjukhusets journalsystem.

De programmeringsspråk som används för applikationen har varit Python och SQL (Structured Query Language). Det är i python tillåtet att ha flera klasser i samma fil vilket gör Python till ett flexibelt språk när det gäller uppdelning och organisering. Detta har varit till av stor betydelse för hur koden organiserats. Då Python är ett av världens mest populära språk finns det otaliga bibliotek att använda sig av. Det finns olika bibliotek för användargränssnitt samt för grafer och diagram vilka har används i projektet. Det är framför allt på grund av dessa anledningar, i kombination med att Python är ett relativt lätt språk att lära sig om man har programmeringsbakgrund i andra språk, som vi valt att använda oss av det.

Programmeringsspråket SQL, Structured Query Language, är ett språk som skiljer sig helt från det annars använda Python i att det är helt specialiserat i att kommunicera och hantera relationella databaser. Det är utformat för att manipulera data i tabellform och utföra operationer som exempelvis att hämta, infoga, uppdatera och ta bort data. I och med att språket är främst inriktat på databashantering gör detta att det kan använda enklare syntax och fokusera på att formulera frågor och kommandon för att hantera datan, också kallat queries. SQL har blivit det ledande query-språket för att komma åt och hantera data lagrade i databaser - specifikt relationsdatabaser.

Det har gjorts flera olika studier på friska människors resultat i de testen som fysioterapeuter gör för att utvärdera en patients nuvarande tillstånd. Resultaten från dessa studier har sedan kunnat användas för att jämföra med patienters resultat och därmed ge en indikation på hur väl patienten har presterat. Vid upprepade tester och jämförelser kan man sedan se vilken effekt rehabiliteringen har på patienten, samt hur lång den estimerade rehabiliteringstiden är. Resultaten från dessa studier

Inlämnat den 15 juni 2023
Emejladress:{al7108ah-s@student.lu.se , je0512fo@student.lu.se}
Teknisk handledare: Josefin Starkhammar, Lunds Tekniska Högskola
Klinisk handledare: Håkan Carlsson, Health Science Center
Physio Log: An application for easier collection of patient and test data for physiotherapists

presenteras på olika sätt vilket leder till olika användning av resultaten. I en artikel som heter *Reference equations for the six-minute walk in healthy adults* [6] presenteras resultaten av en studie där friska personer i olika åldrar med olika vikt, längd och kön fick utföra ett 'Six minutes walking test'. Deras prestation kunde sedan användas för att skapa en ekvation där de tidigare nämnda parametrarna blir viktade med en faktor, adderade eller subtraherade med de andra viktade värdena för att sedan ge ett slutgiltigt tal som korresponderar till just den personens estimerade resultat.

$$(7.57 * height_{cm}) - (5.02 * age) - (1.76 * weight_{kg}) - 309m = 6MWD \quad (1)$$

$$(2.11 * height_{cm}) - (5.28 * age) - (2.29 * weight_{kg}) - 667m = 6MWD \quad (2)$$

Här är ekvation (1) för manliga patienter medan ekvation (2) är för kvinnliga patienter. Patientens estimerade gångsträcka skrivs som 6MWD visar på hur långt en frisk person, med samma attribut som patienten, hade gått. Fysioterapeuterna på Health Science Center i Lund använder ett kalkylark i Excel för att räkna ut dessa ekvationers resultat. Detta kalkylark är förprogrammerat så att endast patientens attribut behöver skrivas in i specifika celler för att den förväntade distansen ska skrivas ut.

I en annan artikel som heter *Adult norms for the Box and Block Test of manual dexterity* [7] presenteras istället resultaten för en liknande studie men för testet 'Box and Block test'. I denna artikel presenteras resultaten, till skillnad från föregående test, i en tabell som beror på patientens kön, ålder samt ifall de är höger- eller vänsterhänta. Det värdet som korresponderar till patientens kön, ålder och dominanta hand visar därmed på det normala antalet kuber som flyttas under en minut. Genom att jämföra patientens resultat med det genomsnittliga värdet i tabellen kan man därmed utvärdera patientens nuvarande status.

Artiklarna *Grip and pinch strength: normative data for adults* [8] och *Comfortable and maximum walking speed of adults aged 20-79 years: reference values and determinants* [9] är upplagda på samma sätt som för föregående artikel i att normalfördelningen för det korresponderande testet, för en viss åldersgrupp och kön, presenteras i tabellform. Dessa tabeller kan därför användas, likt tabellen för Box and block test, för att estimera patientens prestation i jämförelse med det förväntade resultatet för en person med samma attribut.

Efter att ett test har gjorts och ett resultat har erhållits skrivs detta resultat på minst ett sätt. Detta kan dels göras för hand på ett papper som sedan sparas av fysioterapeuten för att det senare ska kunna användas som jämförelse i ett senare stadie i rehabiliteringen. Skillnaden ger en indikation på utvecklingen hos patienten som ett resultat av rehabiliteringen. Detta papper kan antingen sparas fysiskt eller skannas in för att sedan laddas upp i fysioterapeuternas digitala journal. Det är även möjligt att skriva in resultaten i journalen direkt vilket dels tar kortare tid än att skanna in resultatpappret. Båda dessa sätten har dock gemensam brister i att uträkningarna görs i exempelvis Excel medan datan sparas i journalen som inte har något bra sätt att redovisa datan på för att engagera patienten ytterligare. Det enda möjliga är att erhålla resultatet från förra testet, ännu en gång hämta det estimerade resultatet för att sedan kunna jämföra dessa tal med varandra. Detta leder till väldigt mycket jobb för att dels räkna ut och lägga in datan men också för att hämta den igen och presentera den för patienten på ett förståligt sätt. Enligt Håkan Carlsson som är fysioterapeut på Orupssjukhuset i Höör drar sig många fysioterapeuter från att använda dessa tester i deras utvärdering av patienters rehabilitering. Han förklarar att många tycker att det tar för lång tid vilket gör att de istället väljer andra tillvägagångssätt. Han tycker att dessa tester borde användas mycket mer än idag. Han menar på att testerna ger en god bild av i vilket stadie i rehabiliteringen patienten befinner sig samt även kan fungera som ett sätt att följa patientens utveckling och tillfrisknande.

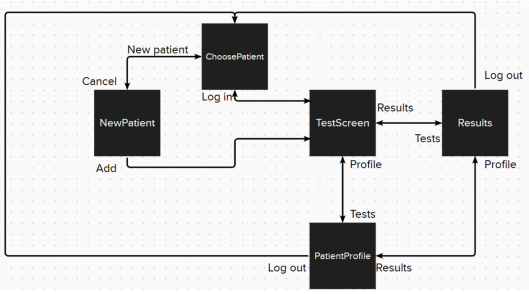
Vi presenterar därför ett program som kommer att lösa alla dessa problemen för fysioterapeuterna. Programmet skapar en egen databas på användarens dator där information kommer kunna sparas. Man kommer därför kunna spara alla patienter med deras attribut. Implementerat i detta program finns även de vanligaste testen som fysioterapeuter gör där användaren kan skriva in det resultat som patienten fick på sitt senaste test. I fall då en ekvation krävs för att räkna ut det resultat som en frisk person med patientens attribut skulle ha presterat, finns även denna integrerad. Resultaten, både patientens prestation men också jämförelsen med det förväntade resultatet kan sedan sparas i samma databas som patientinformationen vilket samlar datan på samma plats. Dessa resultaten kan sedan presenteras i programmet på ett begripligt sätt. Grafer för patientens utveckling kan visas, dels som rådata men också i jämförelse med det förväntade resultatet, men all data kan också visas som en lista som är ordnad efter vilket datum testet lades in på.

Denna rapport är därför fokuserad på hur ett sådant program görs men också optimeras för att göra det så enkelt för användaren som möjligt.

II. Metod

Programmets användargränssnitt är skrivet i Python. Koden har delats upp i filer efter de olika fönstren som användaren ser. Bland annat så finns all kod som har används för att programmera startsidan samlad i en fil. De olika filerna har i sin tur sedan organiserats baserat på vilka delar de oftast interagerar med. Bland annat har alla filer som på något sätt involverar patienternas information samlats i en mapp. De större fönstren som behövdes var en startsida, en för att skapa lägga in en ny patient, en för att välja patient ur en lista, en för att skapa ett nytt test för vald patient, en för att se resultat i form av antingen en graf eller en lista för alla tester gjorda av en viss patient samt en profilsida där patientens uppgifter kan ändras. Utöver dessa behövdes även en rad mindre fönster finnas med i programmet som till exempel felmeddelanden. Dessa mindre fönster låg utanför de mapparna som tillhörde de större fönstren. Det redan gjorda Python-bibliotek som använts till gränssnittet heter 'CustomTkinter' och är en nyare version som baseras på det äldre biblioteket 'Tkinter' som har klasser, metoder och funktioner för att skapa en GUI. 'CustomTkinter'

har ett modernare uteseende out-of-the-box med moduler som t.ex har rundade kanter och mjukare färger. De objekt vars namn innefattar 'CTk' är från 'CustomTkinter'. Om inget annat sägs är de olika delarna av programmet skapade under detta projekts gång. Klassen 'CTkFrame', vilket blev den mest grundläggande beståndsdelen i projektet, är en modul som skapar ett tomt fönster som sedan kan fyllas med olika objekt. Knappar gjordes med klassen 'CTkButton', inmatningsfält skapades med 'CTkEntry' och text med 'CTkLabel'. Med dessa mindre beståndsdelar samt en hel del till kunde till slut flera olika fönster skapas.
För att sedan sammanlänka alla fönster skapades en fil, eller ett script då det körs direkt utan en klass, 'main.py' som höll reda på vilket fönster som för tillfället skulle vara öppet. Fönstren öppnades genom att skapa motsvarande klass och stängdes genom funktionen 'Frame.destroy'. Scriptet 'main.py' var även ansvarig för att se till att programmet över huvud taget startades genom att skapa en 'Toplevel widget' med 'app = ctk.CTk()'. Detta skapade fönstret 'app' där sedan allt som skulle synas kunde lagras i. För att sedan köra programmet och göra det responsivt och kunna ta in användarens input användes i slutet av scriptet 'app.mainloop()'. För att underlätta informationsutbytet mellan dels databasen och själva programmet men också mellan olika fönster i programmet finns klassen 'Patient' och klassen 'Test'. Objekt av denna klass lagrar den inloggade patientens information respektive datan som korresponderar till ett visst test eller testtyp.



Figur 1: En visualisering av hur de olika fönstrena är kopplade till varanda med korresponderande knapparnas namn vid kopplingens början.

Databasen

Ett biblioteket som importerades för databasen är SQLite som fungerar som en brygga mellan Python och SQL. Det går nämligen att använda SQLite i Python-kod och där koppla upp sig mot en databas för att sedan genom denna uppkopplingen kommunicera med databasen på olika sätt. En god egenskap som SQLite har är att ifall ett program försöker koppla upp sig mot en databas som ännu inte finns, kommer denna databasen att skapas. Detta har gjort att vi inte har behövt programmera för flera olika scenarion utan kunde istället fokusera på designen av databasen. Vi har använt oss av olika tabeller i databasen för olika ändamål, en som heter 'Patients' och en som heter 'Tests'. Tabellen Patients innehåller Kolumnerna PatientsID, Name, Age, Weight, Height, Gender och LastOpened där typerna för varje kolumn varierar visas i Tabell I.

Tabell I: De kolumner som finns i Patients med dess korresponderande typ

PatientsID	Name	Age	Weight	Height	Gender	LastOpened
INTEGER	NVARCHAR(50)	INTEGER	INTEGER	INTEGER	NVARCHAR(10)	[Date]

När en patient ska sparas skrivs användarens inmatningar in i programmet över till rätt kolumn i databasen men inte till kolumnen PatientsID. Det talet som kolumnen håller är ett heltal och ökar automatiskt till nästa patient som läggs in i databasen och fungerar som ett unikt id. Detta gör att det kommer vara möjligt att skilja på olika patienter trots att de mot förmodan skulle ha exakt samma attribut. Då detta görs automatiskt av databasen själv kommer det heller inte bli problem ifall programmet startas om. I kolumnen LastOpened skrivs datumet då patienten senast var inloggad in vilket gör det enklare för användaren att hitta rätt patient.

I tabellen Tests finns kolumnerna ID, TestID, PatientsID, Result och Percent och typerna för dessa kolumner varierar i Tabell II.

Tabell II: De kolumner som finns i Tests med dess korresponderande typ

ID	TestID	PatientsID	TestDate	Result	Percent
INTEGER	INTEGER	INTEGER	[DATE]	[REAL]	[REAL]

Kolumnen ID fungerar på samma sätt som det PatientsID som används i tabellen Patients i att det är ett heltal och ökar automatiskt kommer tilldelas testet för att sedan öka till nästa heltal då nya test läggs in vilket gör det möjligt att särskilja olika test från varandra på ett smidigt sätt. TestID kopplas till vilket test som har gjorts, PatientsID till vilken patient som har gjort testet, Result till det resultatet patienten fick på testet samt Percent som är andelen av det förväntade resultatet som patientens resultat är.

Filen som används för att kommunicera med databasen innehåller flera olika funktioner som alla interagerar med databasen på olika sätt. Den första funktionen heter 'createDatabase' och används för att skapa tabellerna inuti databasen med deras korresponderande kolumner. Denna funktion blir kallad varje gång programmet öppnas och därför har vi valt att skapa tabellerna genom SQL-kommandot 'CREATE TABLE IF NOT EXISTS'. Detta kommando kommer skapa tabellerna ifall de inte redan finns i databasen. Vidare finns funktioner som heter 'newPatient' och 'newTest' som fungerar på liknande sätt men där 'newPatient' skapar en ny patient i databasen och newTest skapar ett nytt test. Vid skapandet av en ny patient skickas objektet 'currPatient' in i funktionen. Då 'currPatient' är av typen Patient innehåller det all den information som ska läggas in i databasen. I 'newTest' skickas också 'currPatient' in för att få tillgång till patientens id men även ett Test-objekt skickas in för informationen som ska in i kolumnen TestID. Patientens resultat samt kvoten av patientens resultat delat med det förväntade resultatet skickas in och läggs sedan in i databasen. Funktionen 'oldPatient' gör det möjligt att hämta information om en viss patient från databasen och

spara denna informationen i objektet 'currPatient' så att all information finns lätttillgängligt i delen av koden som körs med Python. Skulle datan för en viss patient behöva ändras av användaren finns funktionen 'editPatient'. Den kallas på när användaren har gjort en förändring till objektet 'currPatient' och uppdaterar då patientinformationen i databasen för just den patienten. Slutligen finns det två så kallade get-funktioner, 'getResults', 'getPercent' och 'getAllPatients'. Dessa används för att hämta resultaten samt procentsatsen för en viss patient i ett visst test respektive för att hämta alla patienterna som finns i databasen. Dessa olika funktioner är sedan implementerade i koden som bygger upp de olika fönstren som programmet består av.

ChoosePatient

Det första fönstret som syns när man startar programmet är en den som heter 'ChoosePatient'. Här möts användaren med ett 'Welcome to Physio Log' längst upp och sedan får man två olika val. Det ena är en lista med alla patienter och det andra är att lägga till en ny patient. Listan gjordes genom att först hämta alla patienter från databasen, skapa en ny lista i Python och lägga in de patienterna i den listan, och sedan lägga in listan i ett fönster där man kan skrolla ifall listan blir för lång. Fönstret gjordes i en annan fil 'RadioButton' där 'ScrollableRadiobuttonFrame' fanns. I denna klass finns metoderna 'add item', och 'get checked item'. Metoden 'add item' Ansvarar för att lägga till ett objekt och sedan visualiersa den som en markerbar cirkel med patientens namn och datum då patienten lades till bredvid. När man skapar en instans av klassen skickar man in listan med patienterna, sedan används en for loop i konstruktorn för att för varje element i listan visa det i användargränssnittet med 'add item'.

NewPatient

I detta fönster finns en titel högst upp med texten 'Patient information' och under denna finns fem inmatningsfält. Till vänster om varje inmatningsfält finns rubriker för varje inmatningsfält och längst ner i fönstrets hörn finns knappar som har texten 'Back' respektive 'Add'. Inmatningsfältens rubriker korresponderar till patientens attribut (namn, ålder, vikt, höjd och kön) som krävs för att göra beräkningarna för testen vid senare tillfälle. I de första fyra inmatningsfälten, som har typen CTkEntry, får användaren själv skriva in patientens attribut medan det sista inmatningsfältet är av typen CTkComboBox. I denna kan användaren välja mellan två förinställda värden, 'Male' och 'Female', då testen kräver att ett av dessa två stämmer in på patienten. Då användaren skriver patientens attribut i inmatningsfälten och trycker på knappen med texten 'Add' kommer den informationen att dels läggas inuti objektet 'currPatient' för lagring av datan och simpel extrahering vid användning av olika funktioner i programmet. Informationen kommer även att läggas in i databasen genom den tidigare nämnda funktionen 'newPatient' och patienten kommer därmed också tilldelas ett patient-id som också lagras i objektet 'currPatient'. Fönstret ändras sedan till att bli fönstret med alla test, TestScreen, och NewPatient-fönstret förstörs. Skulle användaren istället trycka på knappen med texten 'Back' ändras fönstret tillbaka till ChoosePatient.

TestScreen

TestScreen är uppbyggt av en titel, en meny och en CTkScrollableFrame som likt ScrollableRadiobuttonFrame gör det möjligt att presentera text eller objekt inuti ett fönster som är skrollbart. I detta mindre fönster placeras ännu mindre CTkFrame-objekt som ska hålla dels testets namn men också en knapp som sedan ska ta en vidare till just det testets individuella fönster. Genom att spara testens namn i en lista och göra en for loop som går igenom denna skapas ett objekt per test, med tillhörande titel och knapp. Knappen för ett visst test tar en sedan till en ny CTkFrame som läggs ovanpå förra fönstret. Beroende på om det förväntade resultatet för ett specifikt test räknas ut i programmet eller ifall detta läses av från en tabell ser detta fönster lite olika ut. I båda fallen består det unika fönstret av titeln på testet samt ett inmatningsfält med rubriken 'Result' i vilken patientens resultat för testet ska skrivas in i. I de två nedre hörnen finns också två knappar, en som har texten 'Back' och som tar en tillbaka till fönstret med alla testen, samt en knapp med texten 'Save' som sparar testet och tar en vidare till nästa fönster. I fallet då programmet själv räknar ut det förväntade resultatet genom en ekvation är detta hela fönstret men då det förväntade resultatet ska läsas av från en tabell finns ännu ett inmatningsfält. Detta med en rubrik som säger 'Expected result' där det förväntade resultatet ska skrivas in. Då användaren trycker på 'Save' kommer funktionen 'newTest' att köras och lägga in den inmatade informationen i databasen och spara ändringarna. När testet har sparats möts man av ett nytt fönster som har titeln 'Test saved succesfully!' för att indikera att allt har gått rätt till. Med detta presenteras även den informationen som har sparats i databasen vilket innefattar patientens namn, resultatet för testet, resultatets procentandel av det förväntade resultatet samt datumet då testet sparades (dagens datum). Med detta finns även en knapp längst ner som har texten 'Ok!' och som då uppdaterar sidan till att gå tillbaka till fönstret med alla testen.

Menu

Menyn som används i TestScreen används i flera olika fönster i programmet och ser likadan ut oberoende av vilket fönster som är aktivt för att göra den så intuitiv som möjligt. Den aktiveras genom att trycka på en knapp uppe i högra hörnet av det aktiva fönstret som har texten 'Menu'. När denna trycks kallas en funktion som består av en rekursiv loop. Denna loop uppdaterar ett CTkFrame-objekts position med 0.005 pixlar varannan millisekund tills detta objekt, som tidigare låg utanför fönstret, sedan har flyttats in till programmets synfält. I detta objekt finns sedan fyra olika knappar som har texterna 'Profile', 'New Test', 'Results' och 'Log out'. Då knappen 'Profile' trycks kommer man till fönstret 'PatientProfile', 'New test' till 'TestScreen', 'Results' till 'Results' och 'Log out' tar en till 'ChoosePatient'.

Results

Detta fönster är väldigt likt TestScreen i det att titeln på fönstret, 'Results', är placerad ovanför en CTkScrollableFrame

med CTkFrame-objekt för varje test inuti sig. Även menyn finns i det övre högre hörnet likt i TestScreen. Dock är objekten för varje test olika i att de här tar upp hela bredden av det skrollbara fönstret. Inuti objektet finns en titel med namnet på testet samt tre knappar med texterna 'Show chart of percentage', 'Show chart of results' samt 'show list of results'. Genom att trycka på den första knappen hämtas alla procentsatser som korresponderar till den testtypen som är vald samt den patienten som är inloggad. Detta görs genom funktionen 'getPercent' som kommer uppdatera en lista i klassen Test som heter 'percent' med alla procentsatser samtidigt som även en lista som heter 'dates' blir uppdaterad med de korresponderande datumen. Genom att sedan importera pyplot, som är ett grafbibliotek för Python, kan sedan ett figurfönster startas med grafen för där procentsatserna är en funktion av datumen. Nästa knapp, som har texten 'Show chart of results', fungerar på precis samma sätt som föregående knapp men hämtar patientens resultat istället för procentsatserna från databasen. Även denna kommer skapa ett figurfönster där nu resultatet är en funktion av datumet. Vid aktivering av den sista knappen görs ett nytt fönster som har testets namn som titel och har en CTkScrollableFrame i sig som i sin tur presenterar alla tester som gjorts för den testtypen. Denna listan är ordnad efter när testet sparades med det äldsta testet skrivet högst upp i listan. Bredvid testets datum presenteras även patientens resultat för just det testet samt vilken procentsats det korresponderar till. Under det skrollbara fönstret finns en knapp med texten 'Back' som tar en tillbaka till den ursprungliga resultatsidan med alla testobjekt.

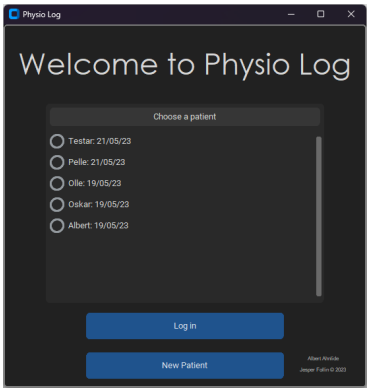
PatientProfile

I Patientprofile, som man kommer till via Menu som det skrivits om tidigare, finns all patientinformation som skrivits in när patienten lades in för först gången. Detta innefattar namn, ålder, vikt, höjd och kön. Det finns även två knappar, 'back' och 'save'. Information syns ovanför varandra på skärmen med text i form av en CTkLabel och ett inmatningsfält i form av en CTkEntry för varje attribut. I CTkEntryn kan användaren skriva in ny information ifall någon justering av patientdatan behövs göras. Texten som skrivs in sparas då i en StringVar och sedan kan man få ut den inskrivna texten genom att använda StringVar.get(). Detta utnyttjas då man klickar på 'save' längst nere till höger. Det som händer då kan delas upp i tre delar. Först går programmet igenom en check för att se till att inte någon av texterna i fältet bryter mot några regler. Texten får inte vara tom. För ålder, vikt och höjd måste texten vara ett tal, alltså en int eller float. Efter detta sparas alla värdena i en instans av objektet 'currPatient'. Denna klass används som en temporär patient som raderas så fort man loggar ut den patienten. Här skrivs all data som användaren ändrat in och sedan skickas hela patientobjektet till databasen som letar upp rätt patient, läser av informationen, och ändrar på den. Sist stängs det nuvarande fönstret ner och istället öppnas det fönster man var innan man klickade på PatientProfile.

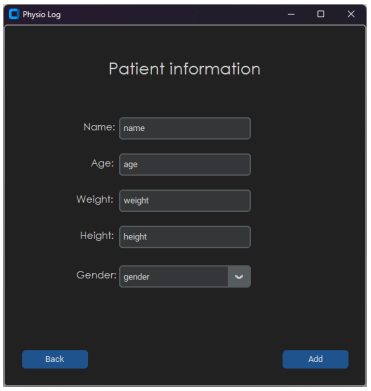
ErrorMessages

För alla de potentiella fel som användaren kan göra finns det felmeddelanden som ser till att programmet inte krashar, fryser eller på något annat sätt hindrar användaren från att använda programmet utan bekymmer. Detta gjordes i klassen ErrorMessages. Detta är en subklass av CTkFrame vilket betyder att ett fönster kommer skapas när man skapar en instans av klassen. I det fönstret finns en rubrik som kort beskriver problemet som uppstått och en mindre, lite längre text som beskriver vad som behövs göras för att rätta till problemet. Längst ner finns en knapp med texten Ok' som stänger fönstret. Rubriken och texten väljer man själv efter man har skapat instansen. I metoden 'setUp' skickar man in rubriken och texten som argument. ErrorMessages användes i nästan varje fönster och är den enda typen av felhantering för användaren som använts.

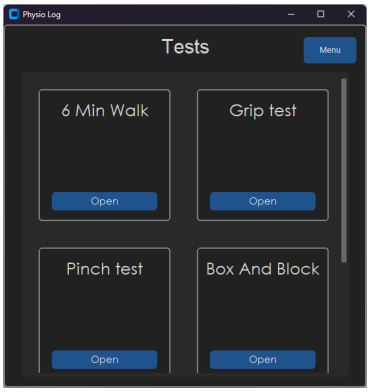
III. Resultat



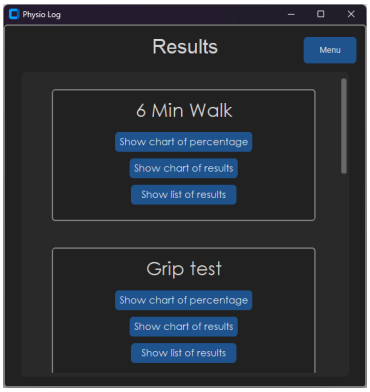
Figur 2: Fönstret ChoosePatient där användaren får valet att logga in som en patient som redan finns i databasen eller gå till fönstret NewPatient för att skapa en ny patient



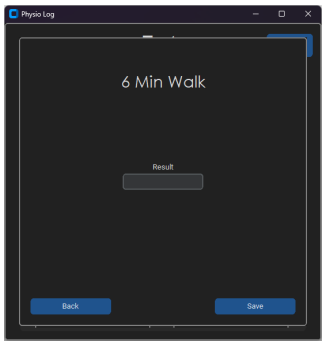
Figur 3: Fönstret NewPatient där användaren, efter inmatning av patientens attribut, kan spara patienten i databasen



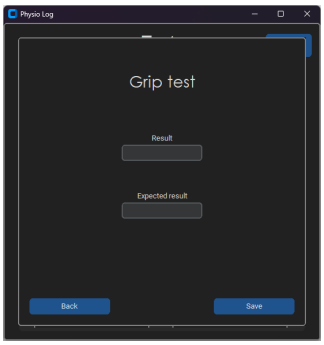
Figur 4: Fönstret TestScreen med alla tester



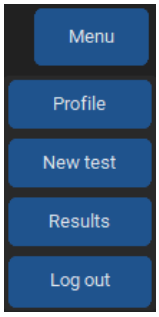
Figur 9: Fönstret Results där resultaten för varje test kan visas



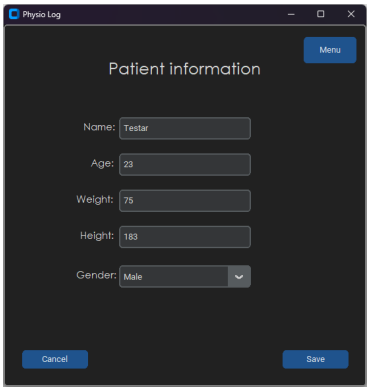
Figur 5: Bild på fallet då det förväntade resultatet räknas ut av programmet



Figur 6: Bild på fallet då det förväntade resultatet läses av i en extern tabell



Figur 7: En bild på hur menyn ser ut med dess knappar



Figur 8: Fönstret PatientProfile där användaren kan ändra patientens information

IV. Diskussion

Vår applikation har klara likheter med det nuvarande systemet som används av fysioterapeuterna i Lund. Båda handlar om att skriva in testresultat från patienter och att kunna få ut en procentsats, åtminstone för 6 min walk test. Målet med vår lösning var att ersätta det excelark som idag används och då måste alla de befintliga funktionerna följa med. Det vår lösning har som inte den nuvarande har är en möjlighet att samla alla olika tester och alla patienters data på samma plats. Detta gör det mycket smidigare och sparar tid för fysioterapeuten. Detta hoppas vi kommer leda till att fysioterapeuter inte ska dra sig från att använda tester för att utvärdera rehabiliteringen. Dessutom finns det möjlighet att visa utvecklingen för patienten i en graf. Detta gör det enklare för patienten att förstå sin egen rehabiliteringsutveckling vilket kan leda till att patienten blir mer engagerad vilket i sin tur resulterar i snabbare återhämtning [10].

Hållbar Utveckling

Vår applikation tillåter användaren att spara all data digitalt vilket, till skillnad från tidigare, gör att man helt och hållet kan undvika användningen av papper. Den som använder programmet kommer dock att spendera mer tid vid datorn då det inte finns en version för telefon eller surfplatta och på så sätt går det åt mer energi vilket, om den energin inte kommer från en förnybar källa, har en negativ effekt på miljön. En annan aspekt är att då det med vårt program är lättare att göra tester finns chansen att testen utförs allt oftare. Det är idag väldigt olika hur flitigt testerna används. En del fysioterapeuter använder det väldigt mycket medan andra mer sällan. På många sätt är det såklart bra om de som i dag inte använder testerna så mycket börjar göra det men samtidigt kräver en del tester ’verktyg’ för att fungera. Som ett exempel kräver Box and Blocks stoppur, två lådor och en hel del mindre kuber. Allt detta måste ju tillverkas på något sätt så här finns en risk för en icke hållbar konsumtion av material.

Etik

I programmet kommer endast initialer att användas för patienternas uppgifter då det inte är ett journalsystem på det sätt som finns inom sjukvården. Detta i kombination med att programmet inte kan kopplas upp till internet gör att personlig

data lagras säkert. Den enda verkliga risken är om någon skulle göra inbrott på den plats där datorn med programmet finns och ta datorn med databasen sparad på. Detta är en risk som inte vi som utvecklar programmet kan göra något åt. Många av dagens datorer har även krypterade hårddiskar för att motverka sådana risker men ifall själva datorn inte är lösenordsskyddad kan datan läcka. Att lösenordssäkra applikationen är inte heller en bra ide då databasen med patientdatan ligger utanför programmet och kan kommas åt ändå. Utöver detta ser vi inga etiska aspekter att ta hänsyn till.

V. Slutsatser

Efter att ha testat programmet stämmer vår tes vi hade från början. Programmet kommer förhoppningsvis att spara tid och vara ett smidigt verktyg för fysioterapeuterna att använda i sin vardag. Det har i sin enkla utformning troligtvis en kort inlärningsperiod. Förhoppningarna är att detta program kommer hjälpa många fysioterapeuter. Troligtvis kommer detta leda till att patienter, som med programmets hjälp kan få en bättre överblick över sin egen rehabilitering, kan återhämta sig snabbare.

VI. Efterord

Vi vill tacka alla våra tekniska handledare på LTH för all hjälp och vägledning vi fått på vägen genom detta projekt. Vi vill speciellt ge ett tack till Håkan Carlsson, vår kliniska handledare som har gett oss både material att jobba med men även tips och svar på alla våra frågor från projektets start till slut. Både Jesper och Albert har bidragit lika mycket till arbetet men på lite olika sätt. Generellt sätt så jobbade båda med användargränssnittet i Python i början för att sedan låta Albert börja med databasen i SQL medan Jesper arbetade vidare i Python. När sedan databasen var klar återgick Albert till Python och tillsammans fulbordade de projektet till ett färdigt program. Rapporten har sedan skrivits av båda genom att dela upp styckena sinsemellan.

Referenser

[1] Socialsytrelsen, ’Statistik om stroke’ https://www.socialstyrelsen.se/statistik-och-data/statistik/alla-statistikamnen/stroke/ 2018-11-30
[2] Akamediska sjukhuset, ’Vård vid stroke’ https://www.akademiska.se/for-vardgivare/sektioner/fysioterapi/behandlingsriktlinjer/lokal-tillampning-av-nationella-riktlinjer-for-vard-vid-stroke/
[3] Physiopeida, ’Six Minute Walk Test’ https://www.physio-pedia.com/Six_Minute_Walk_Test_/_6_Minute_Walk_Test
[4] Physiopeida, ’Pinch Grip Test’ https://www.physio-pedia.com/Pinch_Grip_Test
[5] Physiopeida, ’Box and Block Test’ https://www.physio-pedia.com/Box_and_Block_Test
[6] Enright PL, Sherrill DL. ’Reference equations for the six-minute walk in healthy adults’ *Am J Respir Crit Care Med* 1998 Nov;158(5 Pt 1):1384-7
[7] Mathiowetz V, Volland G, Kashman N, Weber K. ’Adult norms for the Box and Block Test of manual dexterity’ *Am J Occup Ther* 1985 Jun;39(6):386-91
[8] Mathiowetz V, Kashman N, Volland G, Weber K, Dowe M, Rogers S. ’Grip and pinch strength: normative data for adults’ *Arch Phys Med Rehabil* 1985 Feb;66(2):69-74
[9] Bohannon RW. ’Comfortable and maximum walking speed of adults aged 20-79 years: reference values and determinants’ *Age Ageing* 1997 Jan;26(1):15-9
[10] Hibbard JH ’Patient activation and the use of information to support informed health decisions’ *Patient Education and Counseling* 2017 Jan;100(1):5-7